Developing an open-source dashboard for visualisation and analysis of crime data

Adilson Pacheco

Department of Civil, Environmental and Geomatic Engineering,
University College London

Supervisor:
Professor Tao Cheng

A thesis submitted in partial fulfilment of the requirements for the degree of
Master of Science in Geospatial Sciences,
University College London

02 September 2019

CIVIL, ENVIRONMENTAL & GEOMATIC ENGINEERING

Student Name: ADILSON PACHECO (BLOCK CAPITALS)

Programme: MSc Geospatial Sciences *(e.g. MSc GIS)*

Supervisor: Professor Tao Cheng

Dissertation Title:

Developing an open-source dashboard for visualisation and analysis of crime data

DECLARATION OF OWNERSHIP

- I confirm that I have read and understood the guidelines on plagiarism, that I understand the meaning of plagiarism and that I may be penalised for submitting work that has been plagiarised.
- I declare that all material presented in the accompanying work is entirely my own work except where explicitly and individually indicated and that all sources used in its preparation and all quotations are clearly cited.
- I have submitted an electronic copy of the project report through Moodle/turnitin.

Should this statement prove to be untrue, I recognise the right of the Board of Examiners to recommend what action should be taken in line with UCL's regulations.

Signature:	Date:

Abstract

Web-based platforms such as dashboards play a vital role in providing new insights from large data and in presenting the appropriate visualisation of the data in a human-interaction friendly format. With nearly 90% of the world's data being created in the last two years, it is evident that there are great incentives to find cost-effective solutions to process, analyse and visualise this data. Current proprietary visualisation software available to address this issue are still significantly expensive to deploy. Therefore, the use of open-source software has emerged as an alternative to proprietary software. This study focuses on the development of an open-source web platform to visualise and analyse crime data using the city of Chicago as a case study. We used JavaScript and HTML as our primary scripting language, where JavaScript was used to manipulate the DOM elements and handle all the data processing tasks, HTML and CSS were used to set the layout of the webpage. The interaction between client, server and database followed a three-tier architecture. We also adopted a waterfall software model development lifecycle. This model consists of six phases with each phase output acting as input for the next phase with no overlap between phases. The final product resulted in a web-based platform comprising a map, graphs and a filter tab. This platform enables filtering of the data according to input parameters, map visualisation of the data using a choropleth map, a heatmap or a cluster map and analysis of trends and tendencies using a time-series graph and a distribution bar chart. Lastly, we compare our results feature by feature with the leading proprietary visualisation software Tableau. In this project, we show that with the appropriate coding skills it is possible to develop an open-source application that replicates basic features and functionalities available in current proprietary crime dashboards within five weeks. However, when considering developing an open-source application as a solution to a problem, one might first consider time, the complexity and scale of the problem to address. This because open-source applications have proven to be more effective in solving small problems where a tailored solution is required.

Keywords: Open-source, data analysis, data visualisation, JavaScript, HTML

Acknowledgements

I Would like to thank my supervisor Professor Tao Cheng for her great support and guidance in this project and for always making me believe that it is possible. I would like to thank my friends and my family for their unconditional support in all moments of my life. Special thank you to Cardlane Ltd.'s team who always provided the necessary support for me to succeed during my stay in the UK.

Table of Contents

List of Figures

List of Tables

List of Acronyms

API – Application Programming Interface

AJAX – Asynchronous JavaScript And Xml

CSS – Cascading Style Sheets

DOM – Document Object Model

GIS – Geographic Information System

GUI – Graphical User Interface

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

HTTPS – HyperText Transfer Protocol Secure

JSON – JavaScript Object Notation

URL – Uniform Resource Locator

XML – eXtensible Markup Language

Chapter 1 Introduction

1.1 Background

Modern web-based applications allow information to be stored and displayed in real-time from a web browser at almost the same efficiency as a native application. Law enforcement agencies have taken advantage of these technologies to improve crime-fighting strategies as well as to disseminate crime information to the general public. Currently, most law enforcement agencies use commercial products such as ESRI, Tableau and Power BI. However, these commercial tools are usually expensive for medium to small law enforcement agencies. This can also be a major concern in developing countries where police forces might be allocated only a small amount of resources to fight crime. There is an increasing need to find more cost-effective solutions that can provide reliable results. To tackle this issue, we built an open-source web-based application to visualise and analyse crime data. This web application is deployed in JavaScript using Hypertext Markup Language (HTML) and Cascading Style Sheet (CSS). Providing an open-source solution for crime mapping will enable it to be adapted to any region or country in the world and at any geographical scale. Additionally, people from different background and appreciate skills can contribute to further improve the application. Crime is a global issue that hinders social and economic development in parts of the world, providing such a solution at little to no cost can have a significant impact on many communities in the world. This because crime mapping often leads to increased communication between police and citizens (Crowder et al., 2018).

1.2 Research Objectives

This paper outlines the approach, the technology and the data used to create an open-source web-based crime dashboard. To achieve this, we set the minimum requirement for the project to be as follow:

1. Develop an open-source dashboard using JavaScript frameworks and libraries.

2. Include filtering parameters to the data and display the data using a map and graphical visualisation.

3. Find ways of displaying the data in an interactive manner such as adding cross filter functions.

1.3 Thesis Organisation

The remainder of this paper is organised as follows. In chapter 2, a brief literature review discussing the history of crime mapping and analysis, the transition of crime mapping to the digital world, approaches for open-source web-based crime mapping (e.g. software and user interface design) and the current gap in open source web-based crime mapping. Chapter 3 explores the fundamentals of web architecture giving an insight into the technologies operating behind web-based applications illustrating the client-server side interaction. Chapter 4, we present the method of software development lifecycle model, in particular, the implementation of the waterfall software development model as our development method for this project. Chapter 5, we present the results based on the testing phase of the waterfall model for software development. Chapter 6 discusses the results addressing the benefits of using an open-source development, a comparison between our dashboard and Tableau's crime dashboards available in the market and the limitations faced in the dashboard implementation. In Chapter 7, we conclude and provide some recommendation for future works.

Chapter 2 Literature Review

2.1 History of Crime Mapping and Analysis

Location is a component that has always been at the centre of all of the police work. For this reason, maps have always been the preferred ways of depicting crime scenes. According to (Chainey and Ratcliffe, 2005), address or location of the crime scene has been an important component of police work for nearly two centuries. Crime mapping had its origin in France in 1829, where Italian geographer Adriano Balbi and French lawyer and amateur statistician Michel Guerry, first mapped the relationship between violent crimes and educational levels (Eikelboom et al., 2017). Balbi and Guerry used a large one-page sheet containing three shaded maps of France with each shaded map representing a two-variable relationship. The first map showed crime against the person, the second crime against properties and the last school instruction. This practice of geographically displaying variable rates (e.g. crime rates) in areas shaded or patterned in proportion to the measurement of the statistical variable being displayed is called choropleth map (Dent, Torguson and Hodler, 2009). Within a few decades, the adoption choropleth maps to display crime rates spread across the rest of Europe and remained one of the most characteristic forms of crime representation to this date.

Crime analysis techniques themselves have since evolved from the simple choropleth map display. Crime analysis involves the use of a variety of techniques and processes to understand the patterns and relationship of crime. As aforementioned, geographical location plays a vital role in the understanding of crime patterns and relationships. The study of crime analysis dates back to the 1970s, where its spatial dimension began to be fully investigated (Georges, 1978). At this time, it was realised that crime could be explained by the analysis of its spatial components. New crime analysis techniques began to emerge, which included identifying patterns and concentrations of crime. These techniques also emphasised exploring the relationships between crime and environmental or socio-economic characteristics. Combing both analysis and GIS mapping techniques have given origin to the field of crime mapping (Chainey and Ratcliffe, 2005).

2.1.1 Web-based Crime Mapping

Modern crime mapping has shifted from being paper-based to the digital world, allowing the dissemination of information to the public realm as well as making it cumbersome to track long-term crime patterns (Eikelboom et al., 2017). This

transition only became possible with the advent of web and GIS technology in the early 60s. The use of geospatial information in web maps has rapidly expanded thanks to the development of web technologies (Veenendaal, Brovelli and Li, 2017). Web mapping is defined as the process of using computer-generated maps from Geographic Information System (GIS) technologies on the internet. According to (Plewe, 2007; Tsou, 2011), web mapping has transited from static map publishing, static web mapping to interactive web mapping. These changes were a response to the huge volumes of geospatial data being generated and captured over the past decade. With more data available better web mapping tools were needed to be developed to display, analyse and model large data sets (Veenendaal, Brovelli and Li, 2017). Web maps have a wide range of applications, from location-based services, species preservation, monitoring urban pollution to fighting crime.

GIS's breakthrough in the early 2000s, made it become an increasingly popular tool for crime mapping and the initiative of mapping analysis for Public Safety program in the United States (Chainey and Ratcliffe, 2005). GIS plays a major role in investigative and preventive police work, growing its role substantially year after year. (Leong and Chan, 2013) stated that a nationwide survey conducted in the United States in 2001 revealed that nearly 70 per cent of large law enforcement agencies uses digital crime mapping. In the same period, just under half per cent of the police force in the United Kingdom was reported to use digital crime mapping. At this point, web-based crime mapping did not entirely fulfil its role of serving the public.

The next big step for online police mapping was to make the maps publicly available, allowing the general public to view crime and sort crime date by date, location, type of crimes and other variables. Crime data was first made publicly available by third-party companies and developers. Websites such as RAIDS online and chicagocrime.org were among the first to allow citizens to freely view local crime activity in cities. With time, more law enforcement agencies adopted this tendency of releasing their data to the public in an attempt to promote government transparency and accountability, and raise public awareness to criminal activity of their area (Quinn, Cooke and Monaghan, 2019; Eikelboom et al., 2017). With this said, it should be highlighted that public crime maps have a social impact and are primarily intended for society's benefit. (Eikelboom et al., 2017) explains that open-source web-based crime mapping improves safety, planning, trust and communication between law enforcement and the public. The implementation of freely available crime data

platform (e.g. RAIDS) has not only made people more aware of the criminal activities in their surrounding but it also incentive the public to engage with the maps. Such is the case of the RAIDS website used by the London police in Canada, which enables the public to submit anonymous “tips” about suspects and vehicles involved in a crime. In spite of all the benefits associated with web-based crime mapping, there are still concerns regarding public perception and interpretation of the crime data. (Chainey and Tompson, 2012) suggested that these issues could arise from poor usage of cartographic principals when publishing the maps. (Arthur, 2011) added to this by stating that crime mapping is not only useless, but it also creates a false sense of fear which in some areas might hinder social-economic development (Eikelboom et al., 2017; Quinn, Cooke and Monaghan, 2019).

2.2 Approaches to Web-based Crime Mapping

Modern open-source web-based applications are deployed using various scripting languages. This section will describe their main software used for open source web development; the user interface design employed in modern web-based crime mapping application as well as mapping and analytical techniques.

2.2.1 Software

Modern open-source web-based applications are deployed using a various scripting language such as R, JavaScript and Python. This section will describe their main application on web development.

2.2.1.1 JavaScript

JavaScript is the language of the web and generally preferred over the other two. It is an interpreted language with event-driven single-thread execution model support by HTML5. This enables JavaScript to take full advantage of the underlying platform through hardware-accelerated browsers. This improves the overall performance of web-based applications in browsers (Verdú, Costa and Pajuelo, 2016).

2.2.1.2 Python

Python uses two frameworks for web development, these are Flask and Django. Django uses third-party packages to operate with technologies such as NoSQL, real-time Internet communication, and new JavaScript practices. Django has become a popular choice many organisations among those are photo-sharing websites such as Instagram and Pinterest (Rubio, 2017). This because Django eases the creation of

complex, database-driven websites. Flask, on the other hand, is an extensible framework providing a solid core with basic service, while other tasks are achieved through extensions that integrate with the core packages (Grinberg, 2018).

2.2.1.3 RShiny

R is an open-source statistical programming language, it uses the Rshiny package to create interactive web applications. Rshiny gained immense popularity due to its straightforward coding to create and deploy web applications. Rshiny eases the creation of simple intuitive user interface with dynamic filters and real-time exploratory analysis. It integrates additional R packages, JavaScript libraries and CSS for customisation (Seal and Wild, 2016). The greatest advantage of using R and Rshiny together is to combine the statistical power of R and the intuitive user interface of Rshiny.

2.2.2 Design

Web-based crime maps come in different designs, but modern most modern crime mapping web application follows the same structure. This structure comprises three elements, a control panel or filter tab, a map and charts. Some example of this approach is shown in Figure 1. The control panel maintains control over the variables input, these can be, crime type, date and location. Dates and location can be searched to various level of granularity. Dates can be filtered according to years, month, weeks and days while locations can be searched based on counties, wards and cities. These inputs enable the user to narrow the focus to a subset of data based on the parameters input. This provides a quick and easy way to search for crime patterns. The map element of the displays the data either as a point or a layer overlaying the city. The chart element displays a crime pattern in time-series, crime count per crime type and box-plots (Chen et al., n.d.). Different crime mapping and analysis techniques are summarised in section 2.2.2.1.

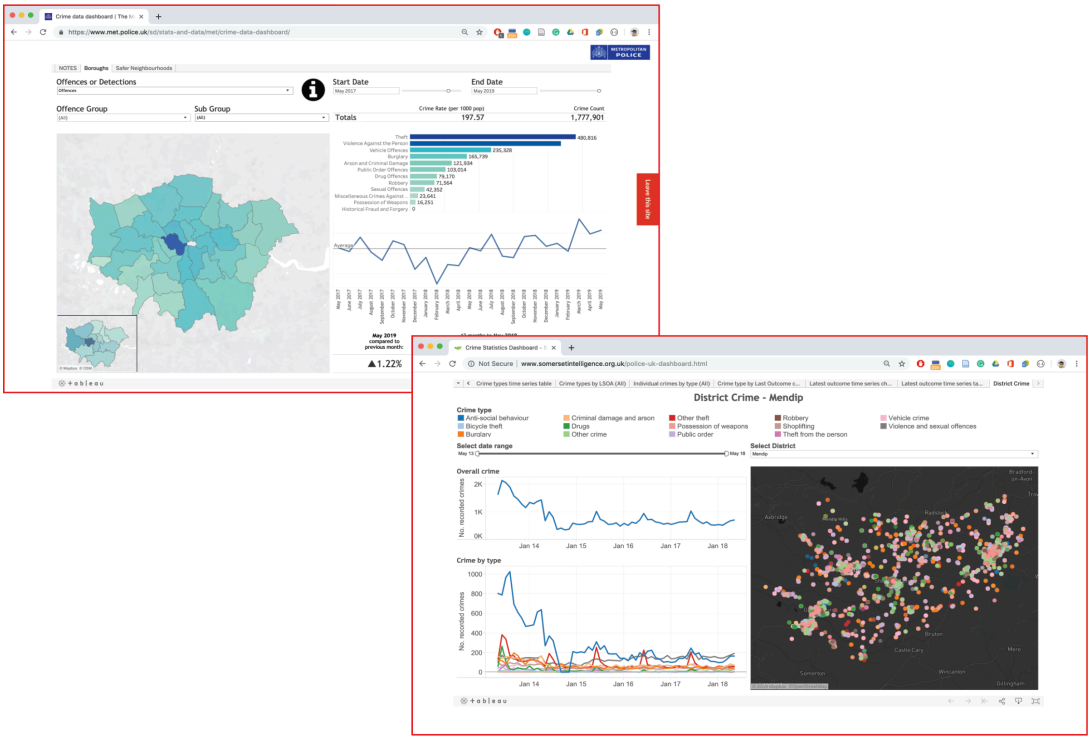

Figure 1: Examples of web mapping applications for crime mapping at the local government level. Both dashboards were deployed in Tableau, the left is the Metropolitan Police dashboard (Metropolitan Police, 2019) and the right is the Somerset Police dashboard (Somerset County Council, 2019).

2.2.2.1 Mapping Techniques Design

As aforementioned, all crimes have spatial component and its spatial distribution is not random. To optimise the process of crime reduction and prevention through the use of web-based applications it is crucial to recognise crime pattern displayed on maps. Maps of crime locations and accurate detection of spatial concentrations or cluster of crime help to identify concentrations crime incident in space and time (Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012). This helps to allocate law enforcement resource at the right place and time. Maps showing areas of concentrations or clusters are called hotspots and these are the preferred crime mapping technique adopted by law enforcement agencies. This because hotspots help predict the locations where future crimes might take place which optimises crime reduction efforts (Chainey and Tompson, 2008). This section describes different hotspots mapping techniques such as choropleth mapping and density mapping applies to crime mapping.

2.2.2.1.1 Choropleth Mapping

Choropleth mapping is a well-known technique to display the spatial distribution of geographic phenomena. This technique is widely used in fields such as epidemiology

and criminology for both reduction and prevention of incidents (Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012). A choropleth map represents an area divided into various geographic areal units (e.g. Boroughs, Wards, Counties, Countries etc). Each areal unit is shaded according to the number of incidents occurred within it. This is a great way of visualising the change in values over a geographical area, which helps in recognising variation or patterns across a study area. Computerised choropleth maps rely on classification algorithm which defines the range of each binding class to group the data into the appropriate class. The accuracy of a choropleth map relies upon choosing the appropriate algorithm, selecting the wrong algorithm may bias the results. Figure 2.2 (b) shows an example of choropleth mapping.

2.2.2.1.2 Grid Mapping

Grid mapping is a form of hotspot mapping that uses uniform grids across a geographical area, and each grid is shaded according to the number of incidents occurring within each of them. Using a uniform grids ease the comparison and identifications of hotspots (Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012). This also overcomes the limitation of choropleth maps caused by the varying size and shapes of the geographic units. Figure 2.2 (c) shows an example of grid mapping.

2.2.2.1.3 Spatial Ellipses Mapping

Spatial Ellipses mapping technique that groups point into clusters based on their proximity. This assumes that objects inside a cluster share similar characteristics to one another and differ from objects in other clusters. The spreading of objects within the cluster is defined by the size and alignment of a standard deviational ellipse fitted to each cluster (Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012). The derivation of spatial ellipse does not depend on defined geographic boundaries, however defining the right parameters is often difficult. Figure 2.2 (d) shows an example of spatial ellipses mapping.

2.2.2.1.4 Kernel Density Mapping

Kernel density mapping is among the most used type of hotspot mapping technique for visualising hotspots in a geographical area. Density maps produce aesthetically appealing maps, which partly explains its popularity. Kernel density maps are generated using a continuous density surface centred on each crime location, density values are estimated at each location by aggregating weights from all density surfaces. This creates a smoothed map representing the point or incident density in

a geographical area (Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012; Chainey, Tompson and Uhlig, 2008). Kernel density maps are used in many academic fields and governmental reports. Figure 2.2 (e) shows an example of kernel density mapping.

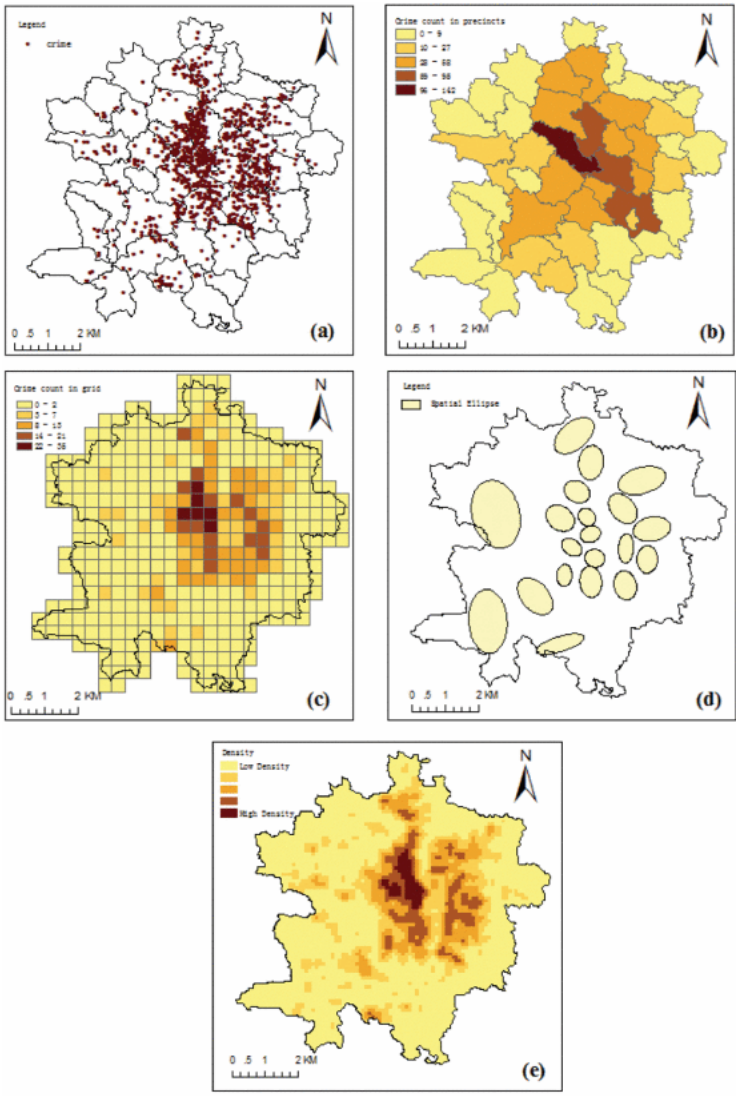

Figure 2: Illustrates different crime hotspot mapping techniques. dot map (a), choropleth map (b), grid map (c), cluster map (d) and kernel density map (e). Source: (Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012)

2.2.2.2 Analytical Techniques Design

Crime analysis consists of conducting several levels of spatio-temporal analysis, which include the examination of long-term trends of crime over several years, by season, and by day of week and time of day. This section will describe different some of the crime analysis display techniques such as time-series and boxplot graph.

2.2.2.2.1 Time-series Graphs

Time-series graphs are visualisation technique is the analysis of data relative to units of time (Santos, 2012). They are displayed as two-dimensional charts of temporal data used to observe phenomena recorded over a period of time (Nason, 2006; Santos, 2012). To create a time-series, we plot the time-increments (e.g. day, week or year) related to a specific location on the x-axis and the measured value or the value of interest on the y-axis. Time-series graphs provide a great feel for trends, for example, stationary or non-stationary, seasonality and periodic fluctuations. This also allows the user to see how often a certain crime occurs within a timeframe.

2.2.2.2.2 Boxplot Graphs

Boxplots graphs are standardised ways of displaying the distribution of a group of numerical data based on a five-number summary: minimum, first quartile (Q1), median, third quartile (Q3) and maximum. The interquartile range (IQR) of a box plot is denoted as a rectangle with the first quartile at one end and the third quartile on the other end, the median is marked at the centre of the rectangle with a vertical line. Horizontal lines extending from the rectangle are called whiskers, which mark two extreme points indicating the overall range. These two extremes are either 1.5 *IQR below the first quartile or 1.5 *IQR above the third quartile. Values beyond these two extreme points are considered outlier with the dataset (Chatfield, 1986). Boxplots are useful tools for comparing several groups of observation distribution.

2.3 Open Source Software

In recent years web-based interactive applications have gained substantial interest in data analysis in all scientific fields. An application can be considered open-source when it distributed with its source code files with a licence that enables the free use, modification and redistribution of the source code (Brunsdon, 2014). All the enhanced and distributed of the application or software carry the same and unique licence as the original to all of them. Open-source applications are generally made available for free. In most case, copyright remains with the creator and attribution to the original and later authors of the source code could also be retained. There are different types of open-source software licences. The most popular types include the General Public License (GPL) and the Berkeley Software Distribution (BSD). The General Public License guaranties that modified versions of the software always be available for further modification and free for use. The BSD asserts copyright enabling redistribution and use of the only if the creator is attributed (Brunsdon, 2014). The

main benefit of open source software is that having access to the source code enables anyone with the appropriate skill set to solve or adapt any limitation or problem encountered in the code (ibid). These enhancements in the code can then be shared with a user community. Developers from different backgrounds can improve a project by approaching it in different ways. This can improve the overall quality by collaboration, consensus and visibility.

2.4 Gap in Open Source Web-based Mapping

When it comes to open source technologies two major benefits are that they free of cost and they are accessible to everyone to improve and redistribute. Although web-based crime mapping applications have been used by law enforcement agencies for some time now, they are still not as ubiquitous as one might think. One of the main reason for this is the cost of implementing commercial proprietary crime mapping software. (Dees, 2003) stated that despite crime mapping software been available for years, they are still fairly expensive. In his paper, (Dees, 2003) compared the affordability of different crime mapping software, which the cheapest at the time been sold for $299, or $229 for an upgrade from the previous version. Fast-forwarding 16 years later and the prices did not get any cheaper. The difference today is that service providers such as Microsoft have in recent years shifted from a one-time payment to a monthly subscription model as a way to yielding faster revenue. Analysing the major proprietary business intelligence and strategy service providers such as Tableau, Power BI and Esri, we have summarised the monthly subscription cost in Table 1. What we can see from these prices is that subscription for crime mapping software are still in fact very expensive and have become more expensive despite the wider range of options available in the market. Due to the high prices practice by commercial mapping software services, these software packages might not be affordable to all law enforcement agencies or police department. (MacEachren, Ross and Roth, 2009; Dees, 2003) agree that most medium to small municipal police department lack crime analytical tools due to its affordability.

Table 1: Monthly use subscription prices for the major commercial web-based business intelligence dashboard services. Source: (ESRI, 2019; Tableau, 2019; Power BI, 2019)

Companies	**Subscription per user/month**
Tableau	US$ 70
Power BI Premium	US$ 4,995
Esri	US$ 50-100

As aforementioned, the second benefit of open source software is its free accessibility to anyone with the appropriate skills to improve and redistribute. Web-based crime mapping is well established in major law agencies in developed countries, however, its use in developing countries is still not widely documented. In some parts of the world, the concept of ePolicing, for instance is a mere fantasy. This problem can be solved partly through the implementation of open-source web applications. An open-source web application for crime mapping could be deployed in any country at region or local scale and more importantly free of cost, and adaptable through code to personal needs. This means that the authors of the application could take the original code and customise it for the needs of the region where the web application will be deployed. We have seen a growing initiative to the use of open-source software in developing countries and they are seen by some as tools that can drive development in different industries for these regions (Dravis, 2003; Weerawarana and Weeratunge, 2004).

Chapter 3 Web Application Architecture

3.1 Fundamentals of Web

In computing, an application is described as software to perform a group of tasks and operations designed for end-user to use. Some application examples include a web browser, a console game, a media player and a photo editor. The first step to develop an application is to identify the appropriate hardware to run the application. Platform applications can be categorised in three types, web-based applications, mobile applications and desktop applications.

3.1.1 Web Architecture

Web-based applications today are entirely three-tier client/server architecture as illustrated in Figure 3. A three-tier architecture is one in which there is an interaction in a client/server environment. The client stores the user interface (UI), the server stores all the logic operations such as processing the browser requests and the server responses. Lastly, the database which stores the data. In a client/server environment, the client communicates with the server using the HyperText Transfer Protocol (HTTP) or HyperText Transfer Protocol Secure (HTTPS). HTTP is the protocol used to access a web server allowing data exchange between the client and server. This protocol enables the user to easily access a web page and perform a search on the World Wide Web (WWW). For example, when the user clicks a hyperlink or types a website name into a web browser, an HTTP request message is sent from the web client to the targeted web server. A request message will consist of a method (e.g. GET or POST) which are operational commands for the server and an URL (Uniform Resource Locator), a set of readable characters to locate the target server(w3schools, 2019d).

Once the request message is sent to the server, it runs an application to process the request and returns an HTTP response message to the client (i.e. the web browser). The web client receives the HTTP response, processes it and renders its content as an HTML page displayed to the client.

This architecture allows each tier (i.e. Web, Server and Database) to be developed and maintain independently on a separate platform. This means that any of the three tiers can be replaced or upgraded in response to technological advancements, for example, the release of a new operating system (OS).

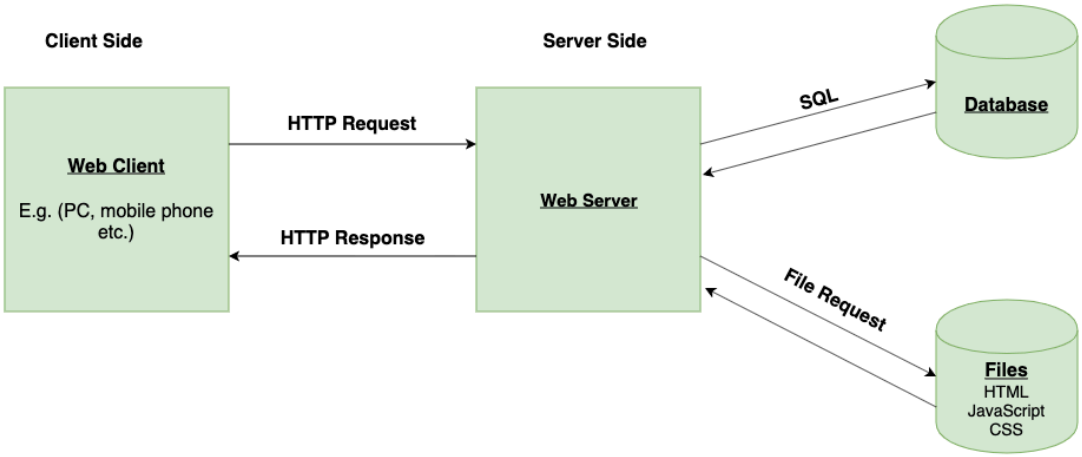

Figure 3. This figure illustrates the three-tier architecture fundamental to all modern web-based application. This architecture comprises a web client, a server and a database. The client/server communicate through an HTTP protocol, where the client-side sends an HTTP request that is then by the server as an HTTP response. The server is responsible for coordinating all logical operations from processing request, querying the database and returning the correct information. The database provides storage of the data.

3.2.1 Client-Side and Server-Side Code

Web application scripts run in the client-side or the server-side also referred to as front-end and back-end. The client-side code refers to any code run on the web browser to improve the visual and operational aspects of the website (Software Engineering Stack Exchange, 2019). The client-side programming relays on three pivotal languages, HTML, CSS and JavaScript.

The server-side code is scripts that run on the server which hosts the website being its main function returning the correct information requested by the browser. In today's web, the main server-side languages are PHP, Ruby on Rail, ASP.NET, Python and C# (MDN Web Docs, 2019c). Server-side code plays a pivotal role in the efficiency and dynamics aspects of a web application affecting the overall user experience.

3.2.1.1 HTML

HyperText Markup Language (HTML) is the standard markup language for websites. HTML consists of a series of elements denoted by tags labels such as paragraph, table, heading etc. Each element is normally surrounded by a pair of angle brackets called opening and closing tags(w3schools, 2019b). The closing tag is denoted by a forward slash inserted before the tag name. HTML elements can have attributes providing additional information about the element. These attributes are always

specified in the opening tag and are written in name/value pairs. Example of attributes is href, height, width etc. The elements in an HTML instruct the browser on how to structure and display the content of a website(w3schools, 2019b).

3.2.1.2 CSS

Cascading Style Sheets (CSS) is a language that describes the style of a document was written in HTML or XML documents (MDN Web Docs, 2019b). CSS applies styles selectively to elements in HTML documents shaping the way they are rendered in the browser. CSS has a structure called ruleset. The rule set consists of individual parts, these are, selector, declaration, proprieties and properties values. Apart from the selector, all other rule sets must be wrapped in curly braces. In the CSS structure, the selector is an HTML element (e.g. paragraph and header) at the start of the rule set indicating the element(s) to be styled. Inside the curly brace is the declaration which states the rule specifying the element's properties to be style, for example, colour, font size etc. Each declaration is separated by a semicolon and comprises a propriety value pair(MDN Web Docs, 2019b). Both CSS and HTML are converted to a common Document Object Model (DOM), an API that defines a logical tree-like structure of the document and the way they can be accessed and manipulated. Using a common structure in both enables CSS styles to be applied to an HTML document.

3.2.1.3 JavaScript

JavaScript (JS) is a cross-platform programming language primarily known as the scripting language for Web pages. However, it is also deployed in many non-browser environments such as Node JS and Adobe Acrobat (MDN Web Docs, 2019a). JavaScript uses the multi-paradigm approach supporting object-oriented, imperative and declarative styles. JavaScript can handle both client and server-side operations. The client-side uses DOM element and regulates the web page behaviour whereas the server-side is responsible for producing a dynamic web page content before the page is displayed or rendered in the browser (MDN Web Docs, 2019a).

3.2.2. Frameworks and Libraries

Frameworks according to (Gizas, Christodoulou and Papatheodorou, 2012) are a library of classes or a collection of function designed to accomplish a multitude of tasks. These tasks vary from managing DOM traversal and manipulations, managing layout, Ajax manipulations, insert visual effects and impose an architecture that provides an unformed way to extend the frameworks such as plug-ins and modules.

For developers, choosing the right frameworks for a project is perhaps the most crucial task at the beginning of a project. This because the frameworks to be applied in a project need first to satisfy the project's need and second, the frameworks used in a project will impact the performance and quality of web applications(Gizas, Christodoulou and Papatheodorou, 2012).

3.2.2.1 jQuery

jQuery is a lightweight, fast and feature-rich JavaScript library. The purpose of jQuery is to simplify complicated HTML document traversal and manipulation, event handling, animation and AJAX with an easy-to-use API that runs across multiple browsers. This means that common tasks that require many lines of JavaScript code can be wrapped into a method called with a single line of code (jQuery, 2019; w3schools, 2019c).

3.2.2.2 Bootstrap

Bootstrap is an open-source front-end framework directed at building responsive web designs, mobile-first sites and template starter page. It contains HTML and CSS based design templates for typography, buttons, tables, navigation and forms, as well as JavaScript-based plugins. Using Bootstraps responsive web design capabilities ensures that developers can create websites that automatically adjust themselves depending on the size of the screen showing the page (Mark Otto, Jacob Thornton, and Bootstrap, 2019; w3schools, 2019a).

3.2.2.3 Leaflet

Leaflet is an open-source JavaScript library used to mobile-friendly interactive maps. It is a cross-platform library which supports mobile and desktop, as well as, HTML and CSS. Leaflet is considered one of the most popular JS mapping libraries along with Google Maps API and OpenLayers. Leaflet enables developers with little to non-GIS background to easily build tiled web maps hosted on a public servant. Additionally, it can load feature data from GIS file format such as GeoJSON files to generate interactive layers, for example, markers and circles with popups when clicked (Leafletjs, 2019).

3.2.2.4 Plotly

Plotly is an open-source high-level, declarative JavaScript charting library for creating graphs and dashboards. Plotly supports basic, statistical, scientific, financial and map

chart, and can also use both D3.js (SVG) and WebGL for graphics rendering. It can load with JSON schema (plotly, 2019).

3.2.3 Databases and Extensions Files

3.2.3.1 API Endpoint

API stands for an application programming interface which is a set of routines, protocols and tools for building software applications. APIs is an interface which has a set of functions that enable developers to access specific features or data of an application, operating system or other devices (Beal, 2019).

3.2.3.2 GeoJSON

GeoJSON is an open standard GIS file format designed for encoding a variety of geographic data structures, along with their non-spatial attributes. A GeoJSON data structure is always an object, meaning that it consists of a collection of name/value pairs called members. GeoJSON supports geographical features such as points, line strings, polygons and multi-part collections. Features in GeoJSON are geometry objects with additional proprieties. A GeoJSON object also stores a coordinate reference system (CRS) which is determined by the crs member. If a crs member is not found nor a parent object's crs can be acquired, a default CRS shall apply to the GeoJSON object. The default CRS is set to WGS84 datum with latitude and longitude units of decimal degrees. GeoJSON features used to represent entities of the physical world, as well as in navigation apps and mobile routing to describe service coverage for instance (Gillies et al., 2019).

Chapter 4 Methods

4.1 Software Development Lifecycle

Software development lifecycle (SDLC) is defined as a set of step-by-step process used to design, develop, and produce high quality, reliable, cost-effective software products within a specific timeframe (Ali, 2017). It essentially defines the way group of developers develop software in the software industry. SDLC models, as it also called, define phases Figure 4 such as requirements, design and development, testing, implementation and deployment (Ali, 2017). There many different SDLC models, namely, Waterfall Model, V-Model, Spiral Model, Iterative Model, Agile Model Big Bang Model, Rapid Application Development Model and software Prototype.

The following section will describe the SDLC model used for the development lifecycle of the open-source dashboard in this project.

4.2 Waterfall Model

The Waterfall model approach was the first widely adopted SDLC model for software development. It is also referred to as a linear-sequential model. The waterfall model uses a linear and sequential flow approach with no overlap meaning that any phase in the development process can only start if the previous phase is completed (Ali, 2017). Simply put, each phase outcome act as input for the next phase.

The Waterfall model breaks down the whole software development process into separate phases. Figure 4 illustrates a graphical representation of the different phases in the Waterfall model. In this model, the next phase can only start when the previous phase is completed with none of the phases overlapping. The sequential phases suggested in the Waterfall model are as follow:

1. **Requirement Analysis**: defines all requirements of the software to be developed and documented in the software requirement specification document at various level of detail.

2. **System Design**: the overall system architecture is defined after studying the requirement specifications.

3. **Implementation**: start of the software development in small programs called units.

4. **Testing**: All small programs are integrated and tested for any fault and failures.

5. **Deployment**: deliver the product to customer or release to the market after the completion of functional and non-functional testing.

6. **Maintenance**: fixe potential issues (e.g. bugs) releasing new patches as well as enhancing releasing a new version (Ali, 2017).

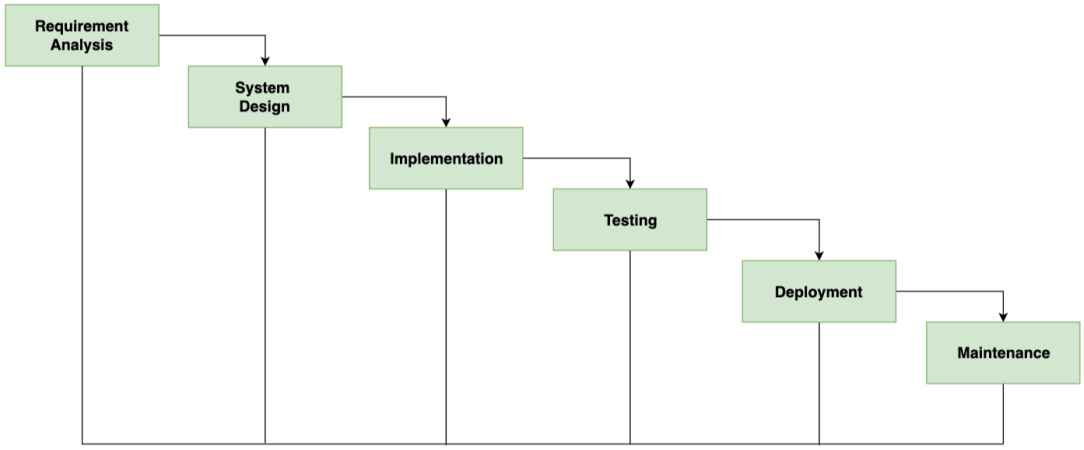

Figure 4: Diagram illustrates a step-by-step process using the Waterfall model for a software development lifecycle. In this model, the whole software development process is divided into separated phases with each phase output acting as input for the next phase, the phases do not overlap. Modified from: (tutorialspoint.com, 2019).

The Waterfall model is not deployable in all software development projects. This approach is only suited to some situations. For examples, short-duration projects with software requirements well documented, clear and fixed as well as the availability of extensive resources with the required expertise to support the software.

There are pros and cons to apply the Waterfall model approach. Some of the advantages are the ease of understanding and use, phases are processed and completed one at the time and each phase has a specific deliverable and review process. The drawbacks, however, are lack of adjustment to frequent project requirement changes, poor performance in large projects and difficult to measure project progress during phases (Ali, 2017).

4.2.1 Requirement Analysis

The requirements set for this project are to develop an open-source web-based dashboard to visualise and analyse crime data. More detailed information regarding the requirements of the project are described in section 1.2.

4.2.2 System Design

This section will address how the requirements from the section above will be achieved. Details of the technologies and the architecture of the system are discussed in chapter 3.

4.2.2.1 Web-technologies

In the previous sections, we have seen that there are a wide variety of web-based technologies available when it comes to web development. There are some key aspects to consider when deciding which hardware or software to choose for a particular project. The first criterium is to understand the magnitude of the project, “is this a big, medium or small size project?”. The next question to be answered is which available tools fully satisfy the requirements of the project. Based on the answer to these questions we can narrow down our tool choices based on their performance and learning complexities of each of them. Taking into account these factors and we have selected the following front-end technologies:

1. HTML5
2. JavaScript
3. CSS
4. Bootstrap
5. JQuery
6. Plotly

With back-end technology of:

1. API Endpoint
2. Local file

We have selected JavaScript as our primary language for this project as it is considered the language of the web. With regards to the frameworks, there is a wealth of frameworks available in JavaScript. However, in this project, we decided to adopt JQuery as our main library. Compare to other frameworks, JQuery is an extremely lightweight fast and feature-rich library. Additionally, it has an easier implementation, extendibility and great browser compatibility supporting cross-browser compatibility. This generally covers the requirement for simple web map application which mainly focuses on map and graph visualisation and does not require complex DOM manipulation. In this project, PlotlyJS was implemented as our graph and charting tools. PlotlyJS is perhaps the easiest open-source JavaScript graphing library to implement, it requires very light coding and has built-in chart functions. It is also a very lightweight library meaning that it renders graphs and charts faster than most other graphing tools. This improves the overall performance of the web dashboard. Our decision to use API Endpoint was due to frequency at which the dataset is update and the ability to perform queries directly to the dataset. Using API Endpoints, it possible to gather data from the latest update.

4.2.2.2 Data Resources

For the accomplishment of this project we have identified three main data resources that are:

1. Chicago Crime API endpoint: https://data.cityofchicago.org/resource/ijzp-q8t2.geojson
2. Chicago Community Area Boundary GeoJSON: https://raw.githubusercontent.com/RandomFractals/ChicagoCrimes/master/data/chicago-community-areas.geojson
3. Chicago Community Area Population JSON file: https://www.chicago.gov/content/dam/city/depts/zlup/Zoning_Main_Page/Publications/Census_2010_Community_Area_Profiles/Census_2010_and_2000_CA_Populations.pdf

Both the Chicago Crime API and the Chicago Community Area Population were obtained from the city of Chicago data portal, and the Chicago Community Area Boundary GeoJSON was sourced from a Github repository. The main reason to use this data set was due to its open-source nature not imposing any restriction on its

usage. With regards to the quality of the data, the city of Chicago data portal is an entity dedicated to promoting access to government data and encouraging development. The stored data is updated and validated frequently ensuring high quality of the data.

4.2.2.2.1 Chicago Crime GeoJSON

The Chicago crime data is sourced in GeoJSON format, which accounted from 2001 onward, excluding the most recent seven days. The GeoJSON contains 22 attributes (columns) and more than 650,000 crime records (rows), with each incident shown at the block level only and specific locations not identified (Chicago data Portal, 2019). The Chicago crime API describes incidents by attributes such as primary type, date, ward, community area latitude and longitude. In Table 2, we have included uniquely the selected attributes relevant to the project. The structure of the data in shown in appendix A1.

Table 2: Description of the selected attributes from the Chicago Crime GeoJSON.

Attribute	**Description**
Date	Date of the incident occurrence or best estimate time. Timestamp at the form of YYYY-MM-DD HH:MIN.
Primary Type	Description of the crime type.
Arrest	Indicates whether an arrest was made.
Domestic	Indicates whether the incident was domestic-related.
Ward	Indicates ward where the incident occurred. Chicago has 50 wards
Community Area	Indicates the community area where the incident occurred. Chicago has 77 community areas.
Latitude	Indicates the latitude of the location where the incident occurred. For partial redaction, this location is shifted from the actual location but falls on the same block.
Longitude	Indicates the longitude of the location where the incident occurred. For partial redaction, this location is shifted from the actual location but falls on the same block.

4.2.2.2.2 Chicago Community Area Boundary

The Chicago community area boundary data is sourced in GeoJSON format. This dataset includes community area boundaries in Chicago from May 2015 onward. This corresponds to the dates when a new City Council is confirmed, in light of the immediately preceding elections. The dataset consists of 9 attributes (columns) and 77 rows each row corresponding to a community area. For this project, we are only using the attributes shown in Table 3. Ward boundaries are defined by a set coordinate in a MultyPolygon data type. The structure of the data in shown in appendix A2.

Table 3: Description of the selected attributes from the Chicago Community Area Boundary GeoJSON.

Attribute	Description
Community	Indicates the community name for each MultiPolygon.
Area_numbe	Indicates the number attributed to each community area.

4.2.2.2.3 Chicago Community Area Population

The Chicago community area population data was converted from a PDF format to a JSON format. This dataset comprises 3 attributes described in Table 4 and 77 rows corresponding to each community area**.** Population size is estimated based on the 2010 census. The structure of the data in shown in appendix A3.

Table 4: Description of the Chicago Community Area Population JSON data.

Attribute	Description
Community_Area_Num	Indicates the number attributed to each community area.
Community_Area	Indicates the community name.
Population_2010	Indicates the population size for each community area based on the 2010 census.

4.2.3 Implementation

4.2.3.1 Data Request

The initial process in this development consists of loading the data resources from the API endpoints and local file. The data is loaded through a jQuery AJAX request. An AJAX stands for Asynchronous JavaScript And XML. The jQuery AJAX method uses a browser built-in XMLHttpRequest object to request data from a web server,

processes the returned data using JavaScript and displays in an HTML DOM. This whole process occurs asynchronously where data is exchanged with a web server in the back-end. This enables to update parts of a web site, without the need to reload the whole page. Figure 5 shows the implantation of AJAX to load the data from the API endpoints and local JSON file.

```
$.ajax({
        url: 'https://data.cityofchicago.org/resource/ijzp-q8t2.geojson',
        method: "GET",
        dataType: "json",
    data: formData,
}).done(function (crime) {
                $.ajax({
                        url:
'https://raw.githubusercontent.com/RandomFractals/ChicagoCrimes/master/data/chicago-community-areas.geojson',
                        type: 'GET',
                        dataType: 'json',
                        data:{
                        },
                }).done(function(boundaries) {
                        $.getJSON("Census_2010_Populations_Chicago.json",
function(data) {
                        }).done(function (population) {});
```

Figure 5: Code snippet illustration data loading from API endpoints and local file.

4.2.3.2 Data Transformation

To display the crime rate into a choropleth map we needed transform the data. This requires merging of the crime data and population data into the boundary GeoJSON (Fig. 6). The crime and the population data were merged into the boundary GeoJSON because the paths (i.e. MultyPolygon) to construct the choropleth layer are generated from this dataset. We created two monster arrays for the crime data and the population data which were in turn bind to the boundary GeoJSON path elements all at the same time. To bind to data sets we need to determine common proprieties on both data sets. In this particular example, we used the community area number which was available in all three data sets. For example, for each community area number in

the Chicago crime GeoJSON, we matched the same number on the Chicago boundary GeoJSON. Then we selected the Chicago crime data value we want to bind and merged it under myBoundary[j]. properties. Value, which can later when plotting the map. A similar process was carried to bind the population data set. Having the crime count and the population size per community area enable the calculation of the crime rate per community area. The structure of the data in shown in appendix A4.

```
var myPop = population;
var myCrime = crime.features;
var myBoundary = boundaries.features;
for(var i =0; i < myCrime.length; i++){
        var dataState = myCrime[i].properties.community_area;
        var dataValue = +myCrime[i].properties.offence;
        for(var k= 0; k < myPop.length; k++){
                var     popArea = myPop[k].Community_Area_Num;
                var popNum = +myPop[k].Population_2010;
                for(var j =0; j < myBoundary.length; j++){
                        var jsonState = myBoundary[j].properties.area_numbe;
                        if(dataState == jsonState){
                                myBoundary[j].properties.value = dataValue;
                                break;
                        }
                        if(popArea == jsonState){
                                myBoundary[j].properties.population_numb =
popNum;
                                var count = myBoundary[j].properties.value;
                                var pop =
myBoundary[j].properties.population_numb
                                myBoundary[j].properties.crime_rate = (count / pop
)*1000;
                                break;
                        }
                }
        }
};
```

Figure 6: Code snippet illustrating data transformation applied to merge data from three JSON dataset.

4.2.3.3 Map and Graph Functionality

The web dashboard contains three different map layers, a choropleth map, a heatmap and a cluster map. Having three distinct forms of map layer enables the user to visualise the data at different levels of granularity. Both heatmap and cluster map enable visualisation from ward level up to the street level as point data. The choropleth layer, on the other hand, is only displayed at the community area level.

The dashboard also contains various graphical displays which are aimed to support the data analysis. Section 5.1.4 describes the graphs and charts used in this project.

Chapter 5 Results

The results presented in this section are displayed according to the requirement of the Waterfall model phases. For the scope of this project, the maintenance phase will be neglected.

5.1 Testing and Deployment

This section will describe the results of the testing in order to validate the system requirements set at the initial phase of the development.

5.1.1 Graphical User Interface

The graphical user interface is an interactive component that enables the interaction between an electronic device and a user. Figure 7 shows the implementation of a user-friendly graphical interface intended to enhance the user experience (UX). The main components of the interface are highlighted in red boxes. These are the sidebar comprising the filtering parameters (e.g. Offence Group and Dates), a map and collapsible buttons that display graphs when clicked.

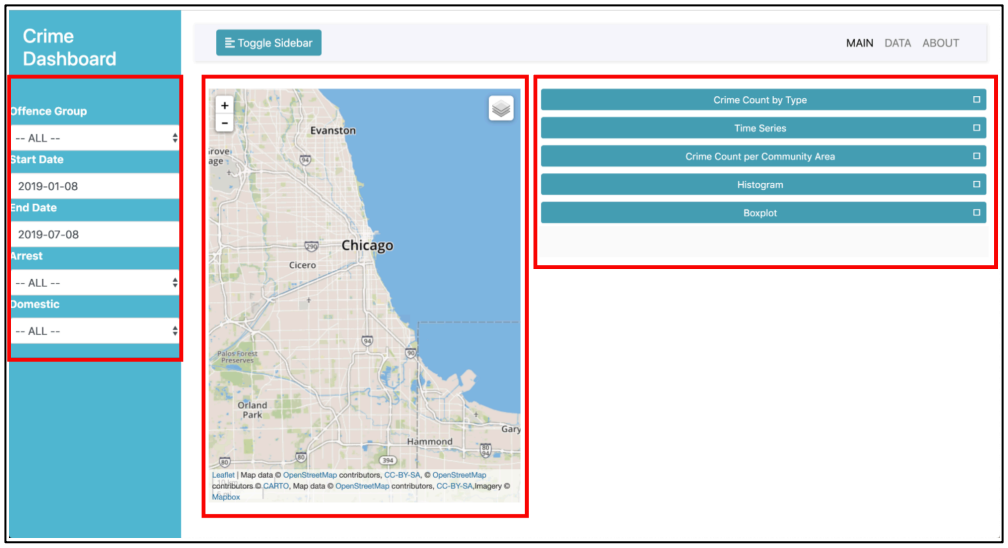

Figure 7: Chicago Crime Dashboard graphical user interface.

5.1.2 Filter Tab

The sidebar (Figure 8) located on the left provides functionalities that enable the user to filter the data in display in the map and the charts. The user can filter the data according to five parameters:

1. Offence Group: dropdown which shows all the distinct crime types in the API

endpoint.

2. Start Date: minimum date at which the incident was recorded in the API endpoint.

3. End Date: maximum date at which the incident was recorded in the API endpoint.

4. Arrest: indicates whether an arrest was made or not using Boolean values true or false and all for selecting both.

5. Domestic: indicates whether an incident was domestic or not using Boolean values true or false and all for selecting both.

Each time the filter parameters are altered, an ajax request is sent to the server which in turns responds with the appropriate to be displayed.

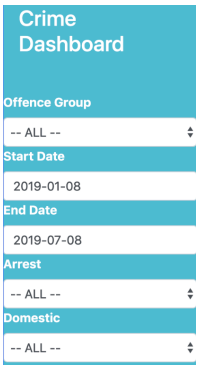

Figure 8: Filter sidebar to set AJAX request parameters for data request.

5.1.3 Maps

In this project, we have used Open Street Map (OSM) and Carto DB basemaps which can be displayed by check the according box in the leaflet layer control located on the top right corner of the map. The basemaps can be overlain by three distinct mapping layers. Three distinct mapping visualisation techniques were adopted (Figure 9), a choropleth map (Figure 9a), a heatmap (Figure 9b) and a cluster map (Figure 9c). A mouse pointing highlight function is available in the choropleth map enabling the user to visualise information dynamically about the chosen community area. This information is displayed in an information box located on the top right corner of the map.

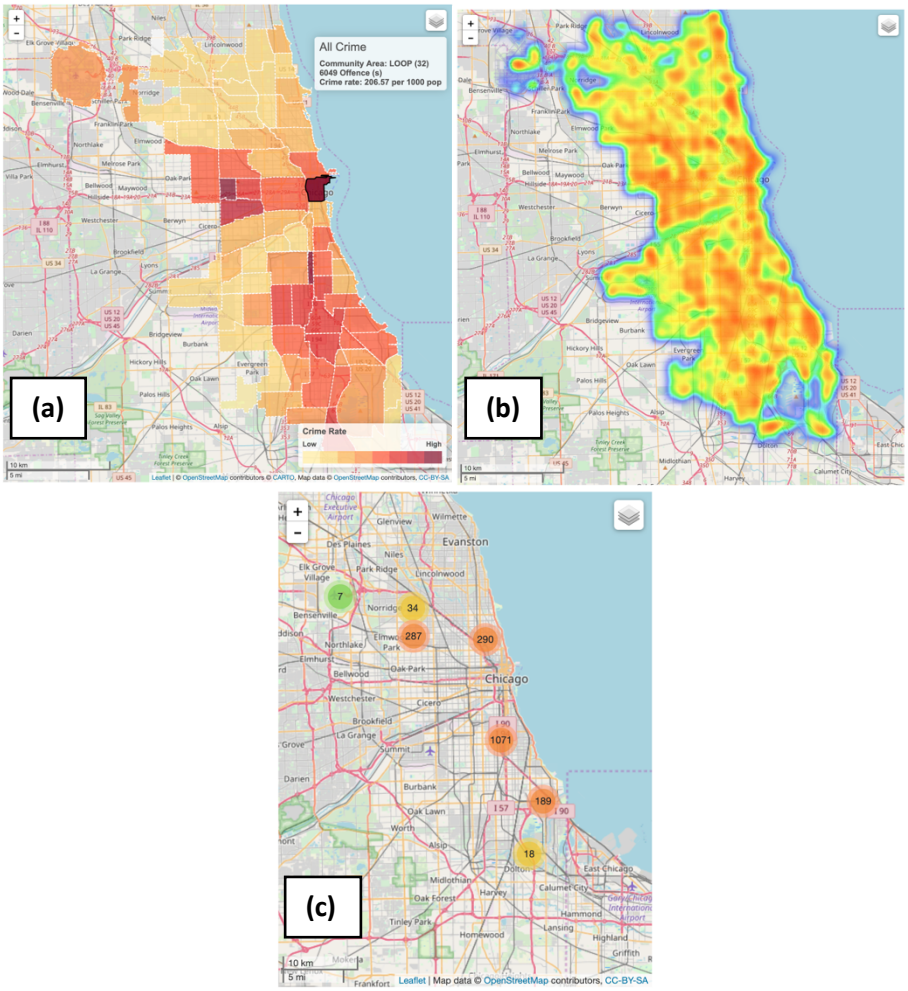

Figure 9: Illustrates different crime mapping techniques implement in web crime platform. (a) Choropleth map, (b) Heatmap and (c) Cluster map.

The choropleth map shows the spatial variation of the crime rate in the Chicago community areas based on the input of the filter variables. To minimise distortion and provide accurate visualisations, we applied k-means breaks classification technique using the one-dimensional k-means clustering algorithm to approximately determine the number of discrete groups of similar values. This arranges each grouping in a way to minimise the variation in each group. The results seen in Figure 9a shows the distinct crime rate per community area with eight binding classes varying from low (pale red-yellowish) to high (darker red) crime rate. By looking at the map (Figure 9a), we can understand that the crime rate is higher in central and south Chicago. Community areas of Loop, Austin, East and North Garfield Park are amongst the area with the higher crime rate. Loop is one of the three downtown community areas in Chicago, therefore, an area where crime intensity is generally higher.

Changing the map visualisation to heatmap or cluster map we can visualise a similar trend but at different spatial resolution. Using the heatmap in Figure 9b, we can zoom in and visualise crime hotspots down to the street. Similarly using the cluster map in Figure 9c, it possible to visualise crime clusters from the city level down to the exact point where the incident occurred. Something to consider, however, is that the point location attributed to a crime incident might not correspond to the exact location where the crime occurred. The recorded location of crime incident is shifted by a few meters to preserve personal and sensitive information.

5.1.4 Charts

In this project, we generated charts using the plotlyJS library. The charts and graphs provided to give the user a different insight into the data besides the map. Four plots are included alongside the map in the main section, a crime count (Figure 10a), a time-series (Figure 10b), an average monthly crime (Figure 10c) and arrest and a domestic bar graph (Figure 10d). Crime count graph shows the total number of incidents recorded within a defined time frame. The time-series illustrates the number of incidents recorded per month within a set timeframe. The average monthly crime count in Figure 10c shows the average distribution of the incidents per month over a defined timeframe. Lastly, the pie charts in Figure 10d show the percentage of arrest and domestic cased for a specific crime within a particular time frame. For example, looking at Figure 10b we can observe a clear seasonal pattern of high and low recorded incidents as well as deducing that have increased over the years. Figure 10c on the other hand, we look at the average monthly distribution of the recorded incidents over a timeframe. From this, we can see that most battery incidents between 2016 to 2019 were on average recorded from May to July.

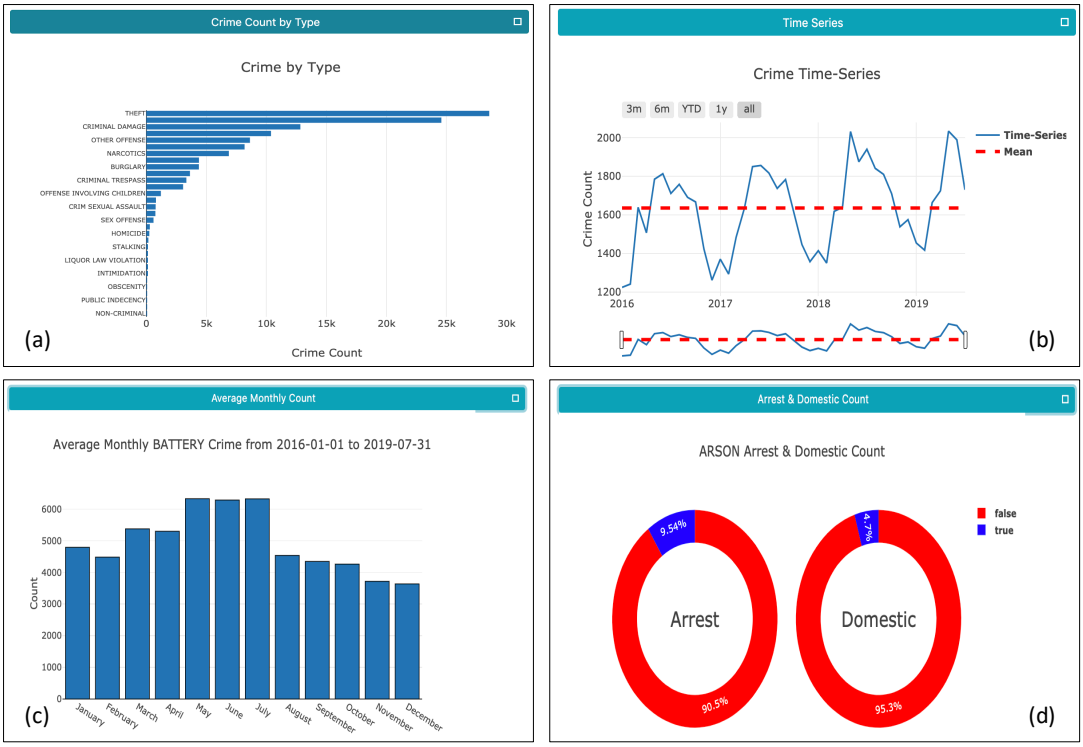

Figure 10: Shows dynamic analytical graphs displaying crime count (a), crime time-series (b), average monthly crime plot (c) and arrest and domestic count (d).

5.1.5 Analysis

The analysis section shown in Figure 11, is a section of the dashboard in which the user can input two different crime type variables under the “Offence Group” dropdown to visualise and compare their trends. Similar to the graphs and charts in the previous section, this section enables the user to carry out a comparative analysis between two crime types identifying whether certain crimes follow similar trends and patterns. The example in Figure 11 compares theft and battery crimes. In this example, we can observe that between the year of 2010 to 2019, both crime types show roughly the same number of recorded incidents 107,673 and 92,327 respectively. Looking at the time series plot for both crime types we can observe that they similar seasonal pattern. Recorded incidents for both crime types have decreased slightly between 2010 and 2015 followed by a gentle rise again between 2015 and 2019. The average monthly recorded crime suggests that while the majority of the battery incident occur between May and July, theft incidents are mostly recorded in July. In terms of arrest and domestic percentage, we can observe that battery incidents have a higher percentage of arrest and domestic-related case, 22.5% and 47.1%, respectively compared to theft incidents with only 10.5% and 2.48% in turns. This comparative

analysis can provide new perceptions of the data. Its user-friendly interaction can be used to communicate essential facts in a large dataset.

Figure 11: shown the analytic section of the dashboard under the analysis tab. In this section, the user can compare statistics between two distinct crime types.

5.1.6 Data

The Data section of the dashboard enables the user to navigate to a new page (Figure 12) where data is displayed in table form. Similar to the main page the user can filter the data using the sidebar on the left. For further filtering of the data, the user can type inside the search space inbuilt in the table as well as ordering the table in ascending and descending order.

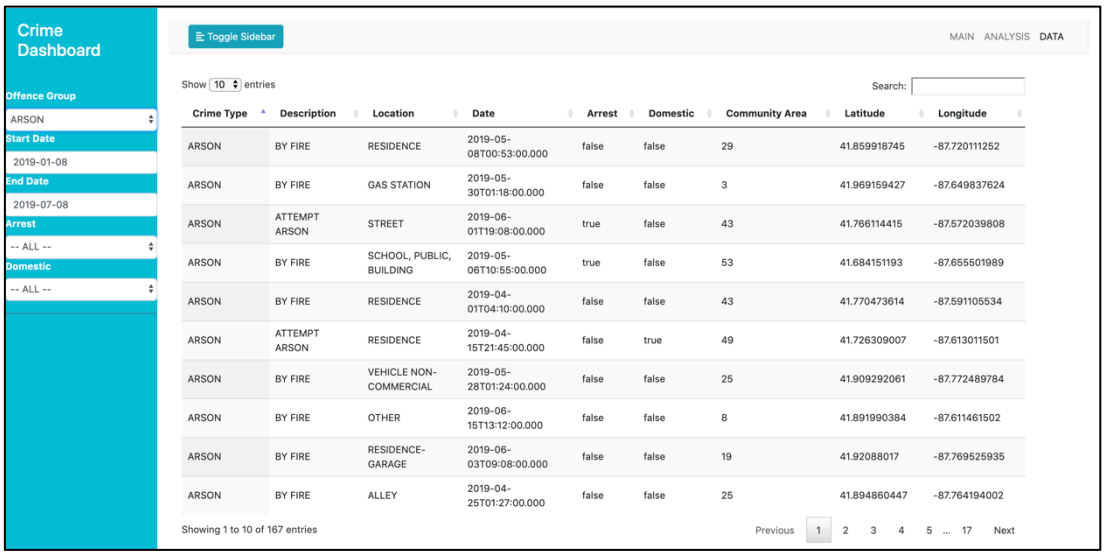

Figure 12: Show crime data presented in table form under the Data section of the dashboard

Chapter 6 Discussion

6.1 Why Open Source?

In recent years, there has been an incredible amount of data been generated daily. In many fields, the analytics of this big data provides a new insight from the data which would otherwise be unnoticed. This new perception of the data is often used to support fast decision making, whether it is in crime-fighting, science or disaster management. Finding human-interaction friendly format of displaying this data is a critical task across many disciplines. Web-based dashboards are the primary solution to this modern age problem. However, as aforementioned, current proprietary web-based dashboards are still highly expensive tools to deploy (Table1). Therefore, there is a great need for cheaper and effective solutions. Open-source web platforms have emerged as more cost-effective solutions to traditional commercial software. For this reason, in this project, we set about to develop an open-source dashboard that enables crime data visualisation and analysis. We will compare our results to the current Metropolitan Police and Somerset County Council Police both developed in Tableau (Figure 1), the leading business intelligence (BI) commercial software.

6.2 Proprietary Vs. Open Source

Comparing the graphical interface design between our open source dashboard and Tableau's crime mapping dashboard. Within 5 weeks of development, we manage to successfully replicate similar user interface features that are typically implemented in any proprietary crime mapping dashboard. These features are a map, a chart and a filtering tab (Figure 7).

In our dashboard map, we have been able to provide multiple map visualisation techniques (Figure 9) providing more dynamic visualisation as well as improving the granularity at which the data is visualised. The user can visualise the data from a regional scale using a choropleth map down to block-level as point data in cluster maps or heatmaps. This contrasts to most crime dashboards deployed in Tableau as these mainly implement one static visualisation technique, either a choropleth or a point map (Figure 1) meaning that the user is constrained to visualise data at one scale only.

Secondly, looking at the graphing aspect of the dashboard. PlotlyJS was the chosen graphing library for this project, the choice for this library is justified in section 4.2.2.1. PlotlyJS library offers interactive graphing and modern analytic tools which resemble

the graphing experience in Tableau. Another factor that made PlotlyJS a more suitable graphing library to use compare to D3.js, for example, is its inbuilt graphing functionality and ease of implementation not requiring heavy scripting knowledge.

Lastly, we compare the data filtering capability between dashboards. Similar to Tableau's dashboards, our dashboard enables data filtering through the variable input using sidebar or tabs by selecting the appropriate parameters in the dropdown (Figure 8). Once the inputs are selected the results are rendered on both map and graphs. However, here also comes one of the limitations that separate Tableau high standard development to our dashboard. Tableau's dashboards are equipped with crossfilter functionality which allows further filtering of the displayed data by selecting elements in the map or graph. Crossfilter capability is a feature that allows the user to drill down the data for deeper insights into the information being analysed. The lack of cross filtering is our dashboard is partially due to the project's time constraint as well as the complexity involved in implementing such functionality to data sets sourced from various location. Table 5 summarises the comparison between our open source dashboard and Tableau's crime dashboard deployed by the Metropolitan Police and the Somerset County Council Police.

Table 5: Comparison of features and functionalities between deployed open-source crime dashboard and Tableau crime dashboard.

Features	**Open Source**	**Tableau**
Map	Able to overlay multiple maps switching on and off dynamically.	Provides mostly single static map layer.
Graphs	High customisable charts and graphs through the use of different available libraries.	Limited to the predefined standard graphs and charts available on the platform.
Crossfilter	Not available.	Crossfilter functionality allowing further filtering of the data set using maps and graphs.

Data Manipulation	Great at manipulating data from various sources simultaneously.	In most cases only manipulates data from a single source.
Coding Requirement	Medium to heavy coding.	Very light coding.

Despite being able to achieve considerable results in developing the dashboard within a short period, time did play an important role in the final result. This raises the question of to what extent is developing an open-source visualisation tool beneficial compared to deploying a proprietary software solution. Open-source software generally only becomes a suitable solution when deployed to solve problems a smaller scale that requires short term solution. This is advantageous to a certain extent because it means that the platform can be tailor to solve specific analytics problem. As pointed out in (Nair, Shetty and Shetty, 2016), the decision to deploy open-source visualisation tools depends on factors such as cost, time for development, dataset volume and expertise. Implementing open-source technology in large projects does often not satisfy the requirements and can be riskier (Ven, Verelst and Mannaert, 2008; Anthes, 2016).

Chapter 7 Conclusions and Recommendations

7.1 Conclusions

In this project, we have illustrated the steps for developing an open-source dashboard for crime data visualisation and analysis as well as comparing our results with proprietary business intelligence applications currently available on the market. We conclude that it is possible to provide similar functionalities as the ones provided by proprietary software such as tableau in a short period of development. Moreover, these functionalities can be tailored to satisfy specific user requirement. The choice of deploying an open-source application over a proprietary application depends on several factors, being time and the scale of the problem to address the main factor to consider. The source code for this application can be accessed on https://github.com/Geobuddy/Crime-Dashboard.

7.2 Recommendations

This project was developed under short time constraints which is the most critical factor to take into account in software development. The maintenance phase of the development model has not been fulfilled due to time constraint. Therefore, one of the first aspect to look at is the user centre design of the application. Carrying out more user testing would enable faster detection of potential bugs in the system which would help to maintain the platform and further improve its usability. Implement spatio-temporal analysis, for example, 3D Kernel density estimation plots for more robust analysis. Additionally, improving the cross-filtering process between the maps and graphs could be of great benefit to usability. Lastly, improve the map layers by aggregating the spatial data to the smallest spatial unity available to improve precision and increase the granularity of the data displayed on the map.

References

Ali, K., 2017. A Study of Software Development Life Cycle Process Models. *International Journal of Advanced Research in Computer Science*, p.10.

Anthes, G., 2016. Open source software no longer optional. *Communications of the ACM*, 59(8), pp.15–17.

Arthur, C., 2011. Crime maps are 'worse than useless', claim developers. *The Guardian*. [online] 2 Feb. Available at: <https://www.theguardian.com/technology/2011/feb/02/uk-crime-maps-developers-unhappy> [Accessed 2 Jul. 2019].

Beal, V., 2019. *What is API - Application Program Interface? Webopedia Definition*. [online] Available at: <https://www.webopedia.com/TERM/A/API.html> [Accessed 31 Aug. 2019].

Brunsdon, C., 2014. *Geocomputation: a practical primer*. 1st edition ed. Los Angeles: SAGE.

Chainey, S. and Ratcliffe, J., 2005. *GIS and Crime Mapping: Chainey/GIS and Crime Mapping*. [online] Chichester, West Sussex, England: John Wiley & Sons, Inc. Available at: <http://doi.wiley.com/10.1002/9781118685181> [Accessed 27 Jun. 2019].

Chainey, S. and Tompson, L. eds., 2008. *Crime mapping case studies: practice and research*. Chichester, England ; Hoboken, NJ: John Wiley & Sons.

Chainey, S. and Tompson, L., 2012. Engagement, Empowerment and Transparency: Publishing Crime Statistics using Online Crime Mapping1. *Policing*, 6(3), pp.228–239.

Chainey, S., Tompson, L. and Uhlig, S., 2008. The Utility of Hotspot Mapping for Predicting Spatial Patterns of Crime. *Security Journal*, 21(1–2), pp.4–28.

Chatfield, C., 1986. Exploratory data analysis. *European Journal of Operational Research*, 23(1), pp.5–13.

Chen, H., Atabakhsh, H., Petersen, T., Schroeder, J., Buetow, T., Chaboya, L., O'Toole, C., Chau, M., Cushna, T., Casey, D. and Huang, Z., n.d. COPLINK: Visualization for Crime Analysis. p.6.

Chicago data Portal, 2019. *Crimes - 2015 | City of Chicago | Data Portal*. [online] Chicago. Available at: <https://data.cityofchicago.org/Public-Safety/Crimes-2015/vwwp-7yr9> [Accessed 17 Jul. 2019].

Crowder, M., Darr, L., Garza, G. and Allen, B., 2018. Opencrimemapping.org: An Online Tool for Visualizing Crime. 1(3), p.19.

Dees, T., 2003. Affordable Crime Mapping. *Law & Order; Wilmette*, 51(12), p.10.

Dent, B.D., Torguson, J. and Hodler, T.W., 2009. *Cartography: thematic map design*. 6th ed ed. New York: McGraw-Hill Higher Education.

Dravis, P., 2003. *Open Source Software: Perspectives for Development*. [online] Washington: The World Bank. Available at: <http://www.infodev.org/files/837_file_Open_Source_Software.pdf>.

Eikelboom, A., Martini, E., Ruiz, L., St. Pierre, A.D. and Tejani, A., 2017. Public Crime Mapping in Canada: Interpreting RAIDS Online. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 52(2), pp.108–115.

ESRI, 2019. *ArcGIS Online User Types | Buy ArcGIS Software Online*. [online] Available at: <https://www.esri.com/en-gb/store/arcgis-online/user-types> [Accessed 13 Jul. 2019].

Georges, D.E., 1978. *The geography of crime and violence: a spatial and ecological perspective*. Resource papers for college geography. Washington: Association of American Geographers.

Gillies, S., Butler, H., Daly, M., Doyle, A. and Schaub, T., 2019. *The GeoJSON Format*. [online] Available at: <https://tools.ietf.org/html/rfc7946> [Accessed 26 Jun. 2019].

Gizas, A., Christodoulou, S. and Papatheodorou, T., 2012. Comparative evaluation of javascript frameworks. In: *Proceedings of the 21st international conference companion on World Wide Web - WWW '12 Companion*. [online] the 21st international conference companion. Lyon, France: ACM Press.p.513. Available at: <http://dl.acm.org/citation.cfm?doid=2187980.2188103> [Accessed 26 Jun. 2019].

Grinberg, M., 2018. *Flask Web Development, 2nd Edition*. 2nd ed. [online] O'Reilly Media, Inc. Available at: <http://proquest.safaribooksonline.com/book/web-development/9781491991725> [Accessed 8 Jul. 2019].

Guiyun Zhou, Jiayuan Lin and Wenfeng Zheng, 2012. A web-based geographical information system for crime mapping and decision support. In: *2012 International Conference on Computational Problem-Solving (ICCP)*. 2012 International Conference on Computational Problem-Solving (ICCP). pp.147–150.

jQuery, 2019. *jQuery*. [online] Available at: <https://jquery.com/> [Accessed 25 Jun. 2019].

Leafletjs, 2019. *Leaflet — an open-source JavaScript library for interactive maps*. [online] Available at: <https://leafletjs.com/> [Accessed 25 Jun. 2019].

Leong, K. and Chan, S.C.F., 2013. A content analysis of web-based crime mapping in the world's top 100 highest GDP cities. *Crime Prevention and Community Safety*, 15(1), pp.1–22.

MacEachren, A.M., Ross, K.R. and Roth, R.E., 2009. GeoVISTACrimeViz GeovisualAnalytics for Spatiotemporal Crime Analysis. [online] Available at: <https://geography.wisc.edu/cartography/people/presentations/RothEtAl_2010_TCIP.pdf> [Accessed 4 Jul. 2019].

Mark Otto, Jacob Thornton, and Bootstrap, 2019. *Bootstrap*. [online] Bootstrap. Available at: <https://getbootstrap.com/> [Accessed 25 Jun. 2019].

MDN Web Docs, 2019a. *About JavaScript*. [online] MDN Web Docs. Available at: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript> [Accessed 24 Jun. 2019].

MDN Web Docs, 2019b. *CSS: Cascading Style Sheets*. [online] MDN Web Docs. Available at: <https://developer.mozilla.org/en-US/docs/Web/CSS> [Accessed 24 Jun. 2019].

MDN Web Docs, 2019c. *Server-side web frameworks*. [online] MDN Web Docs. Available at: <https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Web_frameworks> [Accessed 24 Jun. 2019].

Metropolitan Police, 2019. *Crime data dashboard | The Met*. [online] Available at: <https://www.met.police.uk/sd/stats-and-data/met/crime-data-dashboard/> [Accessed 30 Jun. 2019].

Nair, L.R., Shetty, S.D. and Shetty, S.D., 2016. INTERACTIVE VISUAL ANALYTICS ON BIG DATA: TABLEAU VS D3.JS. 12(4), p.12.

Nason, G., 2006. Stationary and non-stationary time series.

Plewe, B., 2007. Web Cartography in the United States. *Cartography and Geographic Information Science*, 34(2), pp.133–136.

plotly, 2019. *plotly*. [online] Available at: <https://plot.ly/javascript/> [Accessed 25 Jun. 2019].

Power BI, 2019. *Pricing & Product Comparison | Microsoft Power BI*. [online] Available at: <https://powerbi.microsoft.com/en-us/pricing/> [Accessed 13 Jul. 2019].

Quinn, A., Cooke, L. and Monaghan, M., 2019. An exploration of the progress of open crime data: how do ongoing limitations with the Police.uk website restrict a comprehensive understanding of recorded crime? *Policing and Society*, 29(4), pp.455–470.

Rubio, D., 2017. Introduction to the Django Framework. In: *Beginning Django: Web Application Development and Deployment with Python*. [online] Apress. Available at: <http://proquest.safaribooksonline.com/book/web-development/django/9781484227879/1dot-introduction-to-the-django-framework/a441241_1_en_1_chapter_html> [Accessed 8 Jul. 2019].

Santos, R.B., 2012. *Crime Analysis With Crime Mapping*. SAGE.

Seal, A. and Wild, D.J., 2016. Netpredictor: R and Shiny package to perform Drug-Target Bipartite network analysis and prediction of missing links. *bioRxiv*, p.080036.

Software Engineering Stack Exchange, 2019. *web development - What are the differences between server-side and client-side programming?* [online] Software Engineering Stack Exchange. Available at: <https://softwareengineering.stackexchange.com/questions/171203/what-are-the-differences-between-server-side-and-client-side-programming> [Accessed 23 Jun. 2019].

Somerset County Council, 2019. *Dashboard of crimes recorded by Avon & Somerset Police in the Somerset County Council area Source: police.uk data download*. [online] Tableau Software. Available at: <https://public.tableau.com/views/SomersetPolice_ukdashboard/Frontpage?:embed=y&:showVizHome=no&:host_url=https%3A%2F%2Fpublic.tableau.com%2F&:tabs=yes&:toolbar=yes&:animate_transition=yes&:display_static_image=no&:display_spinner=yes&:display_overlay=yes&:display_count=yes&:showTabs=y&:loadOrderID=0> [Accessed 30 Jun. 2019].

Tableau, 2019. *Buy Tableau | Tableau webstore*. [online] Available at: <https://buy.tableau.com/en-gb/teams-orgs> [Accessed 13 Jul. 2019].

Tsou, M.-H., 2011. Revisiting Web Cartography in the United States: the Rise of User-Centered Design. *Cartography and Geographic Information Science*, 38(3), pp.250–257.

tutorialspoint.com, 2019. *SDLC Waterfall Model*. [online] www.tutorialspoint.com. Available at: <https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm> [Accessed 30 Jun. 2019].

Veenendaal, B., Brovelli, M.A. and Li, S., 2017. Review of Web Mapping: Eras, Trends and Directions. *ISPRS International Journal of Geo-Information*, 6(10), p.317.

Ven, K., Verelst, J. and Mannaert, H., 2008. Should You Adopt Open Source Software? *IEEE Software*, 25(3), pp.54–59.

Verdú, J., Costa, J.J. and Pajuelo, A., 2016. *Dynamic web worker pool management for highly parallel javascript web applications*. [online] Concurrency and Computation: Practice and Experience. Available at: <http://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.3739> [Accessed 8 Jul. 2019].

w3schools, 2019a. *Bootstrap 4 Get Started*. [online] Available at: <https://www.w3schools.com/bootstrap4/bootstrap_get_started.asp> [Accessed 25 Jun. 2019].

w3schools, 2019b. *Introduction to HTML*. [online] Available at: <https://www.w3schools.com/html/html_intro.asp> [Accessed 24 Jun. 2019].

w3schools, 2019c. *jQuery Introduction*. [online] Available at: <https://www.w3schools.com/jquery/jquery_intro.asp> [Accessed 25 Jun. 2019].

w3schools, 2019d. *What is HTTP*. [online] Available at: <https://www.w3schools.com/whatis/whatis_http.asp> [Accessed 23 Jun. 2019].

Weerawarana, S. and Weeratunge, J., 2004. *Open source in developing countries*. [online] Stockholm: Sida. Available at: <http://www2.sida.se/shared/jsp/download.jsp?f=SIDA3460en_Open+SourceWEB.pdf&a=3055> [Accessed 5 Jul. 2019].

Appendices

Appendix A1 – Example of GeoJSON format from the Chicago crime API endpoint.


```
{
  type: "FeatureCollection",
  features: [
    {
      type: "Feature",
      geometry: {
        type: "Point",
        coordinates: [
          -87.642992854,
          41.757366519
          ]
        },
      properties: {
        location_state: "",
        location_zip: "",
        x_coordinate: "1172605",
        domestic: false,
        latitude: "41.757366519",
        updated_on: "2018-02-10T15:50:01.000",
        description: "TO VEHICLE",
        location_address: "",
        :@computed_region_6mkv_f3dw: "21554",
        arrest: false,
        :@computed_region_d9mm_jgwp: "20",
        location_city: "",
        year: "2015",
        :@computed_region_vrxf_vc4k: "66",
        :@computed_region_d3ds_rm58: "229",
        longitude: "-87.642992854",
        :@computed_region_awaf_s7ux: "17",
        block: "075XX S EMERALD AVE",
        fbi_code: "14",
```

```
        ward: "17",
        id: "10365064",
        date: "2015-12-31T23:59:00.000",
        beat: "0621",
        y_coordinate: "1854931",
        community_area: "68",
        :@computed_region_rpca_8um6: "59",
        location_description: "STREET",
        district: "006",
        iucr: "1320",
        case_number: "HZ100370",
        :@computed_region_bdys_3d7i: "511",
        :@computed_region_43wa_7qmu: "32",
        primary_type: "CRIMINAL DAMAGE"
        }
      },
    }
}
```


Appendix A2 – Example of GeoJSON format from the Chicago community area boundary.


```
{"type":"Feature","properties":{"community":"DOUGLAS","area":"0","shape_area":"46004621.1581","perimeter":"0","area_num_1":"35","area_numbe":"35","comarea_id":"0","comarea":"0","shape_len":"31027.0545098"},"geometry":{"type":"MultiPolygon","coordinates":[[[[-87.60914087617894,41.84469250265398],[-87.60914874757808,41.84466159842403],[-87.6091611204126,41.84458961193954],[-87.60916766215838,41.84451717732316],[-87.60916860600166,41.844456260738305],[-87.60915012199398,41.84423871659811],[-87.60907241249289,41.844194738881015],[-87.60900627147821,41.84410646928696]]]]
```

Appendix A3 – Example of GeoJSON format from the Chicago Community Area Population Census.


```
[
 {
   "Community_Area_Num": 1,
   "Community_Area": "Rogers Park",
   "Population_2010": "54991"
 },
 {
   "Community_Area_Num": 2,
   "Community_Area": "West Ridge",
   "Population_2010": "71942"
 }]
```


Appendix A4 – Example of merge JSON object.


```
0:
geometry:
coordinates: [Array(1)]
type: "MultiPolygon"
__proto__: Object
properties:
area: "0"
area_num_1: "35"
area_numbe: "35"
comarea: "0"
comarea_id: "0"
community: "DOUGLAS"
crime_rate: 0.054830573527799104
perimeter: "0"
population_numb: 18238
shape_area: "46004621.1581"
shape_len: "31027.0545098"
value: 1
__proto__: Object
type: "Feature"
```